CLAIMS

What is claimed is:

1. An extensible rule-based technique for optimizing predicated code, comprising:
    if-converting an abstract internal representation; and
    mapping the if-conversion to a machine representation.

2. The technique of claim 1, further comprising:
    eliminating predicates from the mapped if-conversion.

3. The technique of claim 1, the eliminating of predicates comprising:
    eliminating a predicate defining instruction by interpretation.

4. The technique of claim 1, the eliminating of predicates comprising:
    eliminating a guarding predicate of a safe instruction by speculation.

5. The technique of claim 1, the eliminating of predicates comprising:
    eliminating a guarding predicate of an unsafe instruction by compensation.

6. The technique of claim 1, the eliminating of predicates comprising:
    eliminating a guarding predicate of an unsuitable instruction by reverse if-conversion.

7. The technique of claim 1, further comprising:
    optimizing the machine representation.

13. The apparatus of claim 10, the eliminating of predicates comprising:
    means for eliminating a guarding predicate of a safe instruction by speculation.

14. The apparatus of claim 10, the eliminating of predicates comprising:
    means for eliminating a guarding predicate of an unsafe instruction by compensation.

15. The apparatus of claim 10, the eliminating of predicates comprising:
    means for eliminating a guarding predicate of an unsuitable instruction by reverse if-conversion.

16. The apparatus of claim 10, further comprising:
    means for optimizing the machine representation.

17. An extensible rule-based technique for optimizing predicated code, comprising:
    if-converting an abstract internal representation;
    mapping the if-conversion to a machine representation;
    eliminating predicates from the mapped if-conversion,
    wherein the eliminating of predicates, comprises
        eliminating a predicate defining instruction by interpretation;
        eliminating a guarding predicate of a safe instruction by speculation;
        eliminating a guarding predicate of an unsafe instruction by compensation;

8. An extensible rule-based system for optimizing predicate code, comprising:
    a processor for executing instructions; and
    an instruction for
        defining predicates;
        testing a branch instruction; and
        assigning a defined predicate to the branch instruction based on a result of the test.

9. An extensible rule-based method for optimizing predicate code, comprising:
    defining a predicate;
    testing a branch instruction; and
    selectively assigning the defined predicate to the branch instruction based on a result of the test.

10. An apparatus for optimizing predicate code, comprising:
    means for if-converting an abstract internal representation; and
    means for mapping the if-conversion to machine representation.

11. The apparatus of claim 10, further comprising:
    means for eliminating predicates from the mapped if-conversion.

12. The apparatus of claim 10, the eliminating of predicates comprising:
    means for eliminating a predicate defining instruction by interpretation.

eliminating a guarding predicate of an unsuitable instruction by reverse if-conversion; and

optimizing the machine representation.

18. A technique of supporting predicated execution without explicit predicate hardware, comprising implementing a test branch instruction.

19. The technique of claim 18, wherein the test branch instruction converts a branching condition based on condition codes to Boolean data in a general register so that a full logical instruction set can be used to produce optimal code.

20. A system of supporting predicated execution without explicit predicate hardware, comprising:

a processor for executing instructions; and

an instruction for

converting a branching condition based on condition codes to Boolean data in a general register so that a full logical instruction set produces optimal code; and

guarding a set of instructions unsuitable to speculate enclosed by a branch.

21. A method of supporting predicated execution without explicit predicate hardware, comprising implementing a test branch instruction.

22. The method of claim 22, wherein the test branch instruction converts a branching condition based on condition codes to Boolean data in a general

register so that a full logical instruction set can be used to produce optimal code.

23. An apparatus of supporting predicated execution without explicit predicate hardware, comprising:

means for implementing a test branch instruction; and

means for eliminating predicates using the implemented test branch instruction.